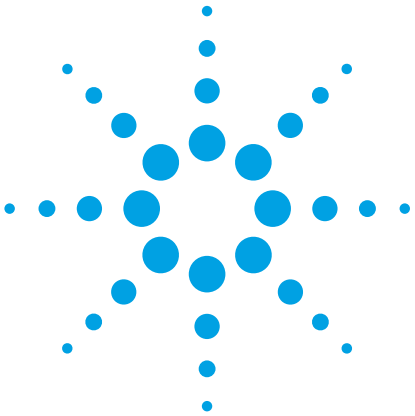


Agilent

VisualBasic VisualC#によるオシロスコープ制御入門

Anticipate ___Accelerate ___Achieve

Agilent Technologies

内容

1. はじめに 3
2. 制御プログラム作成の準備 3
２．１． ソフトウェア環境 3
２．２． ハードウェア環境 4
２．３． 測定器の接続と VISA アドレスの確認 4
3. 基本的なプログラムの作成 6
３．１． プログラムの起動 7
３．２． 参照ライブラリの追加 8
３．２．１． Visual Basic の場合 8
３．２．２． Visual C#の場合 10
３．３． 測定器との通信 11
4. 制御プログラムの基礎知識 15
４．１． コマンド 15
４．２． ライブラリ 16
４．２．１． ライブラリとは 16
４．２．２． ライブラリの利点 16
5. サンプルプログラム 17
５．１． サンプルプログラムの説明 17
５．２． サンプルプログラムの操作方法 18
５．３． サンプルプログラムのコードの確認、カスタマイズ 19
５．４． サンプルプログラムのコマンド 19
6. オシロスコープ制御のポイント 21
６．１． 制御プログラムの流れ 21
６．２． トリガ待ち 22
６．３． 波形のデータ転送 24
６．３．１． ASCII 転送とバイナリ転送 24
６．３．２． バイナリブロックデータ転送 25
６．３．３． バイト、ワード 25
６．３．４． 符号付数値表現 26
６．３．５． データ転送時のバイト順 26
６．３．６． データ読み取り時のタイムアウト設定（よくあるエラー） 26
６．４． 取得したバイナリデータの電圧値への変換 27
６．５． デバッグ 27
7. 測定器プログラムのサポート 32
自動計測関連情報 33

1. はじめに

電圧変化を 1 分ごとに 24 時間連続で見てみたい。複数の測定器を組み合わせて、条件を変化させながら測定していきたい。様々な条件の下での測定や、長時間のデータ取りは、人の力だけでは、多くの時間が費やされてしまいます。こういった場合、PC と測定器を接続し、PC から測定器を制御し、データ取りを行うことが一般的です。

しかし、一昔前までは、GPIB など測定器専用バスラインを構築し、更に、プログラム言語を購入し、その上で、プログラムを作成していく必要があり、技術的に分り難く、また、お金がかかるものでした。しかし、最近の測定器には、LAN、USB の汎用インターフェースが搭載されているものが増えつつあり、また、測定器メーカーからは、わかりやすい通信制御用ライブラリが用意され、プログラム言語も無料で入手できるようになるなど、技術的にも、金銭的にも、敷居が低くなっています。

ここでは、制御プログラムを、これらから作り始める方のために、一般的なプログラムの作成方法を基礎から、更に、実際のオシロスコープ（Agilent InfiniiVision シリーズ）のサンプルプログラムを通して、実践的な部分まで、ご案内させて頂きます。基礎的な部分は、弊社測定器に限らず、他メーカーの測定器を制御する際にも、ご参考になるものと思います。

2. 制御プログラム作成の準備

２．１． ソフトウェア環境

本文章の内容を実施するためには、マイクロソフト社の Windows OS に、以下のものがインストールされている環境が必要です。これらは、インターネット上でも入手・購入することが可能です。なお、Agilent IO Libraries Suite は、弊社測定器ではなくとも、お使いになることができます。

- .Net Framework 2.0 以降がインストールされている Windows OS
- Visual Basic または Visual C# (Express 版でも可)
- Agilent IO Libraries Suite（以下の弊社 Web から無償でダウンロードが可能）
  [IO Libraries Suite ソフトウェアダウンロードサイト]
  http://www.agilent.com/find/iolib
- Sample Program（以下の弊社 Web から無償でダウンロードが可能）
  [ダウンロードサイト]
  http://www.agilent.com/find/iolib

２．２． ハードウェア環境

本章の内容を実施するには、PC や測定器の種類、搭載されているインターフェースにもよりますが、必要に応じて、PC と測定器を接続するためのインターフェースカードや、ケーブル類が必要となります。現在の Agilent 社の測定器には、制御用インターフェースとして、主に、USB、GPIB、LAN、RS232C（シリアル通信）などが搭載されています。個別の測定器に搭載されているインターフェースに関しては、データシートやマニュアルをご確認ください。以下は、オシロスコープに搭載されている制御用インターフェースです。

- [参考] Agilent 社製オシロスコープ搭載 リモート制御用インターフェース

DSOX2000/3000	USB、（オプション）、LAN（オプション）
DSO5000	USB、GPIB、LAN
DSO/MSO6000	USB、GPIB、LAN
DSO/MSO7000	USB、GPIB（オプション）、LAN
DSO/MSO8000	USB、GPIB、LAN
DSO80000	USB、GPIB、LAN
DSO/MSO9000	USB、GPIB（オプション）、LAN
DSO/DSOX90000	USB、GPIB（オプション）、LAN

お使いの測定器に合わせて、制御に用いるインターフェースを検討します。その際、LAN、USB であれば、一般的なインターフェースであるため、PC に搭載されている可能性が高いですが、GPIB は、測定器用インターフェースであるため、一般的な PC に搭載されていません。そのため、GPIB にて制御を実施する場合には、GPIB インターフェース、GPIB ケーブルを、別途、ご購入する必要があります。これらは、弊社にて販売しておりますので、お気軽にお問い合わせください。

２．３． 測定器の接続と VISA アドレスの確認

制御する測定器を指定には、対象となる測定器の VISA アドレスを使います。Agilent IO Libraries Suite の Agilent Connection Expert を使って、VISA アドレスを確認する方法を説明します。

① 測定器を PC と接続します。

ここでは、USB ケーブルで接続したものとして、ご説明を進めます。

LAN で接続する場合は、PC と測定器の IP アドレスの設定/確認が必要です。また、ケーブルの種類にご注意ください。PC と直接接続する場合、通常は、クロスタイプのケーブルが必要です。ハブ等を用いて接続した場合は、一般に使われているストレートタイプのケーブルを使用することができます。

② Agilent Connection Expert を起動します。

スタートボタンから [スタートボタン]>[Agilent IO Libraries Suite]>[Agilent Connection Expert]を実行します。(**エラー! 参照元が見つかりません。**参照)

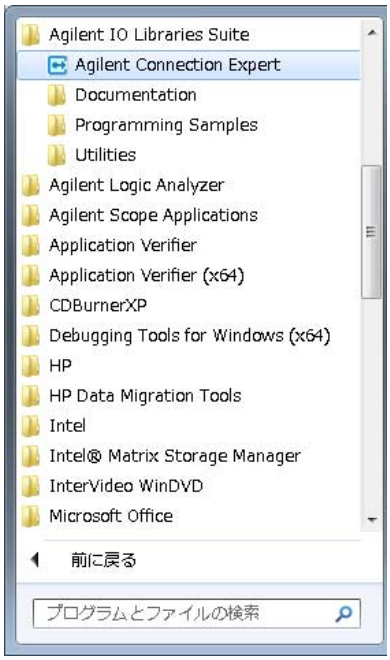


図 1 Connection Expert の起動

③ ケーブルで接続されている測定器は、モデル名の横に、緑色のチェック印が示されます。
もし、「×」、「！」の印が付いている場合は、測定器が認識されていない事を表します。原因として、測定器の電源が入っていない、ケーブルが接続されていない、測定器によっては、測定器側の設定（お使いになるインターフェースの選択）がなされていない事が考えられます。

④ 使用する測定器を選択し、表示される VISA アドレスを確認してください。

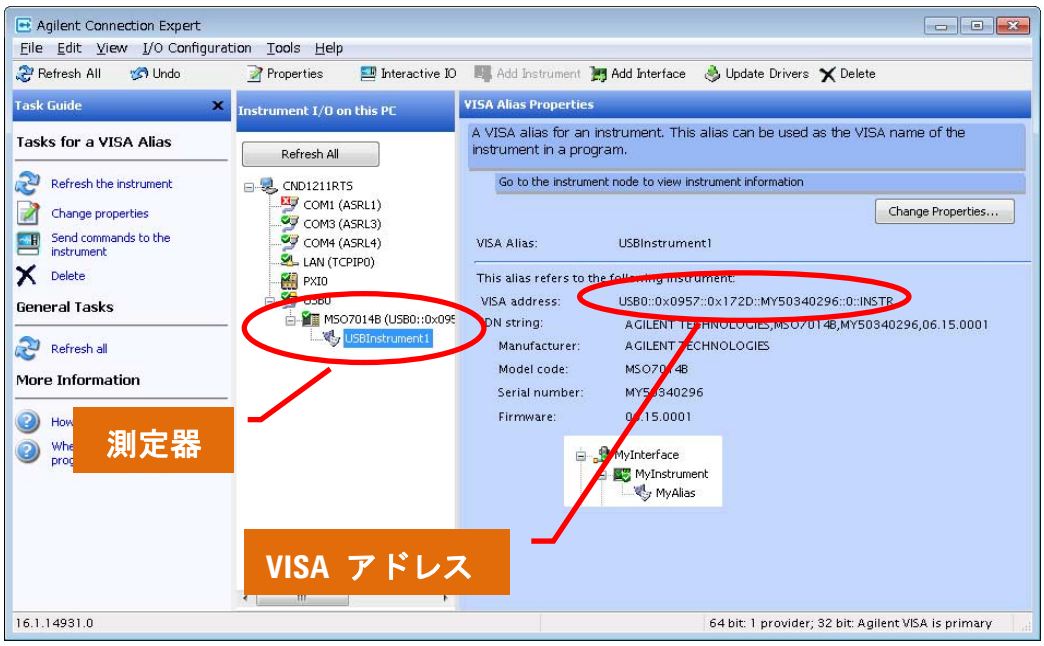


図 2 VISA アドレスの確認

3. 基本的なプログラムの作成

この章では、Windows フォーム アプリケーションの作成を例に、測定器に対して、「*IDN?」という測定器の ID(モデル名)を問う文字列(コマンド)を送り、その応答を得るまでを説明します。「*IDN?」は、IEEE488.2 にて定められている測定器の共通コマンドであり、Agilent、他メーカー、また、オシロ、その他の測定器問わず、殆どの測定器で、ご使用になれます。

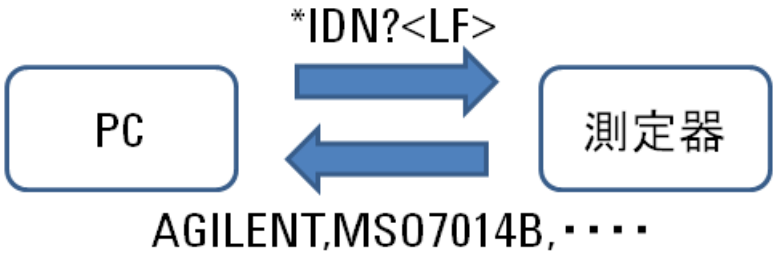


図 3 *IDN?の送信

測定器の ID を問い合わせるプログラムは、測定器制御プログラムを作成する上で、最初に行われる動作確認として、よく利用されています。良く使われる理由としては、測定器にコマンドを送る、測定器からの応答を受け取るといった単純な動作を通じ、

- 測定器と測定器用インターフェースの接続
- 測定用インターフェースと PC との接続
- ライブラリのインストール
- 開発環境の設定

など、測定器制御プログラム作成の前提となる環境が設定できている事を確認できるためです。また、ID は、測定の動作に関係なく、得られることも、理由の一つです。(測定のトリガが掛からなくても、測定器は ID を返す事ができます)

なお、ここで、ご説明する「*IDN?」を送るプログラムは、下記のサイトより、サンプルプログラムとしてダウンロードが可能です。この手順書と合わせて、ご参照ください。

[ダウンロードサイト]
http://www.agilent.co.jp/find/InfiniiVision-sample

３．１． プログラムの起動

各プログラム環境で [ファイル] > [新しいプロジェクト] から Windows フォーム アプリケーションを選択し、デザインの画面を表示してください。

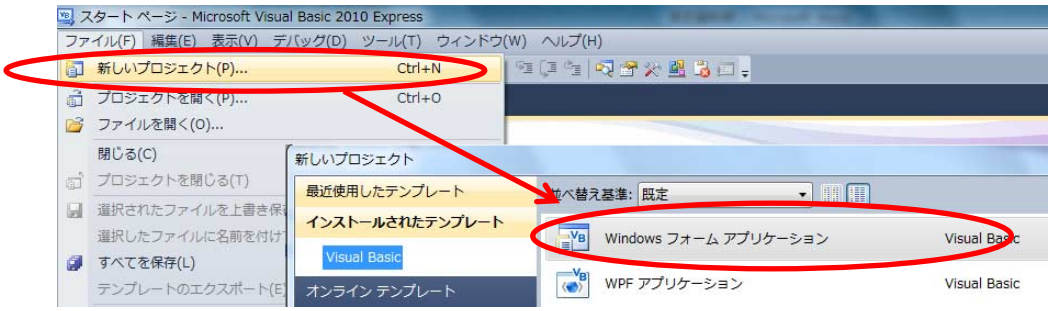


図 4 Visual Basic の場合

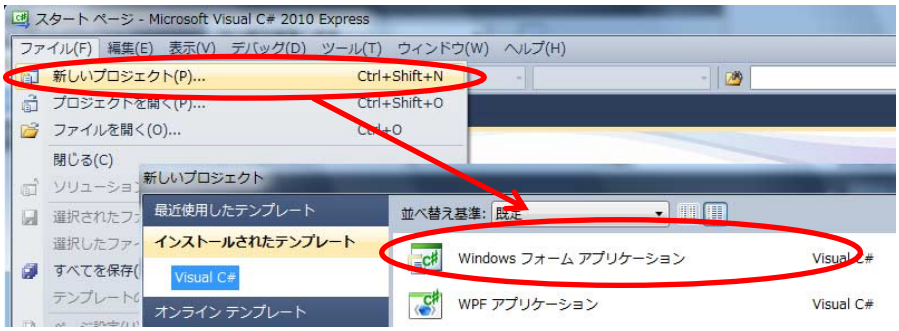


図 5 Visual C#の場合

３．２． 参照ライブラリの追加

プログラムから測定器と通信を行う際には、ライブラリの参照を追加し、それを用いて通信を行います。このライブラリをお使いになるには、IO Libraries Suite をインストールする必要があります。

３．２．１． Visual Basic の場合

① プロジェクトのプロパティを開きます。

ソリューションエクスプローラの My Project をダブルクリックするか(図 6)、メニューバーから[プロジェクト]>[プロパティ (P)]をクリックしてください。

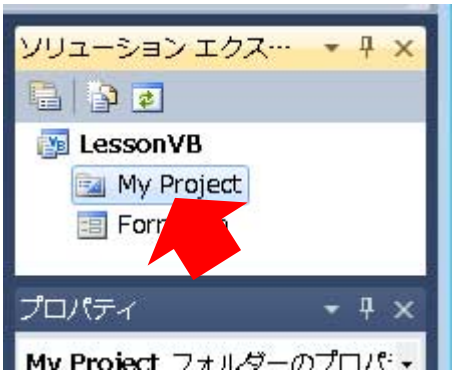


図 6 プロジェクトのプロパティウィンドウを開く(Visual Basic)

② 参照タブの追加をクリックします。(図 7 参照)

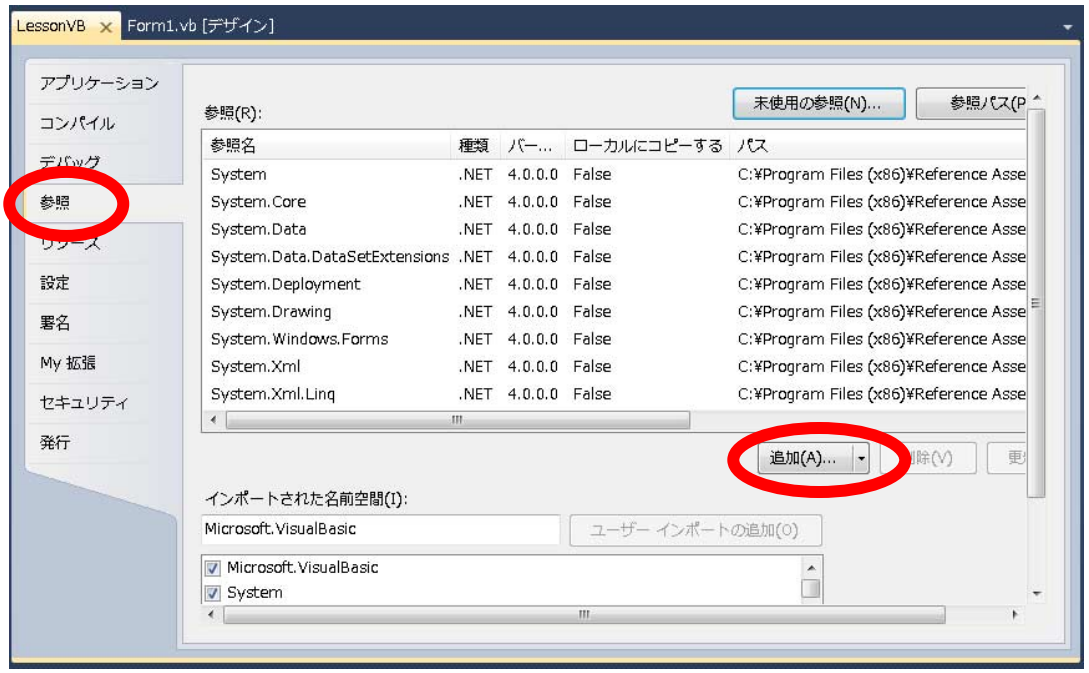


図 7 参照の追加(Visual Basic)

③ COM ライブラリから VISA COM 3.0 Library を選択して OK ボタンを押します。(図 8 参照)

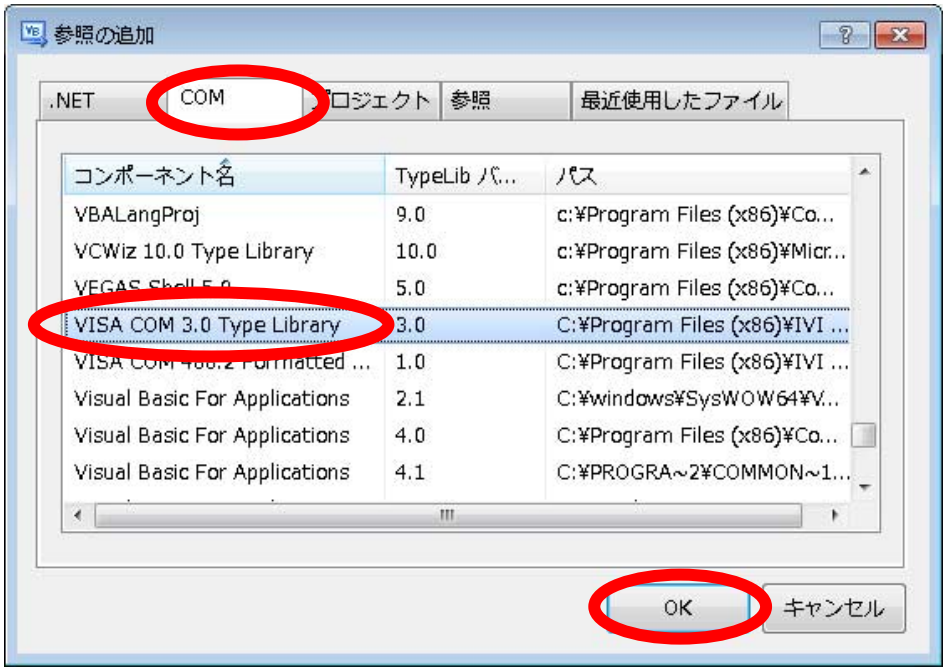


図 8 VISA COM 3.0 Library の追加(Visual Basic)

④ インポートされた名前空間の Ivi.Visa.Interop にチェックを入れます。(図 9 参照)

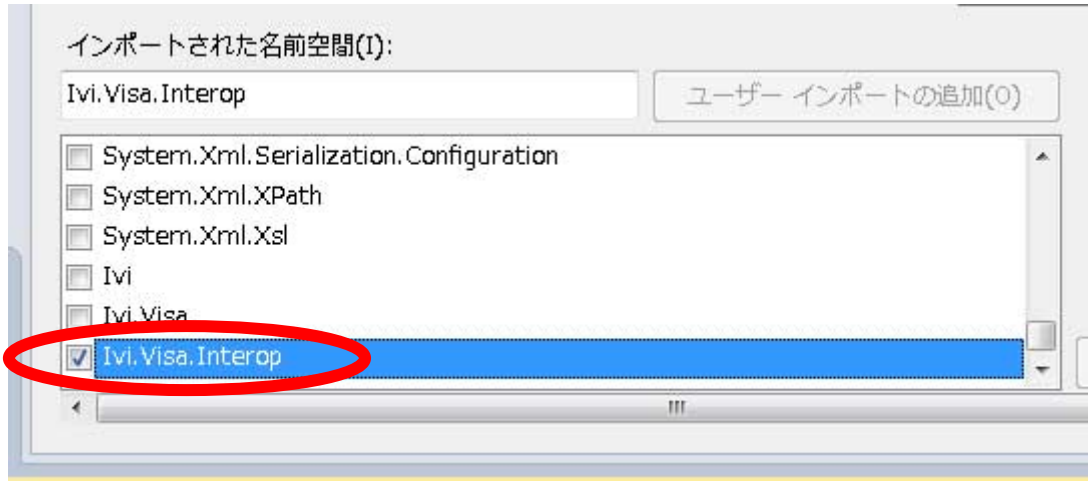


図 9 名前空間の追加(Visual Basic)

３．２．２． Visual C#の場合

⑤ プロジェクトのプロパティを開きます。

ソリューションエクスプローラの My Project をダブルクリックするか(図 6)、メニューバーから[プロジェクト]>[プロパティ (P)]をクリックしてください。

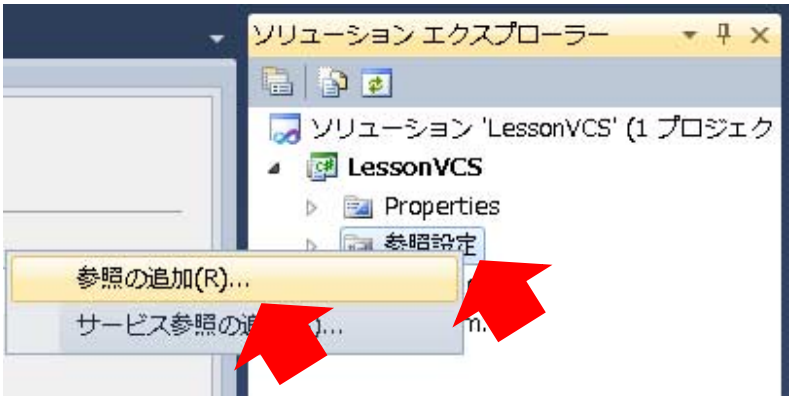


図 10 プロジェクトのプロパティウィンドウを開く(Visual C#)

⑥ COM ライブラリから VISA COM 3.0 Library を選択して OK ボタンを押します。(図 11 参照)

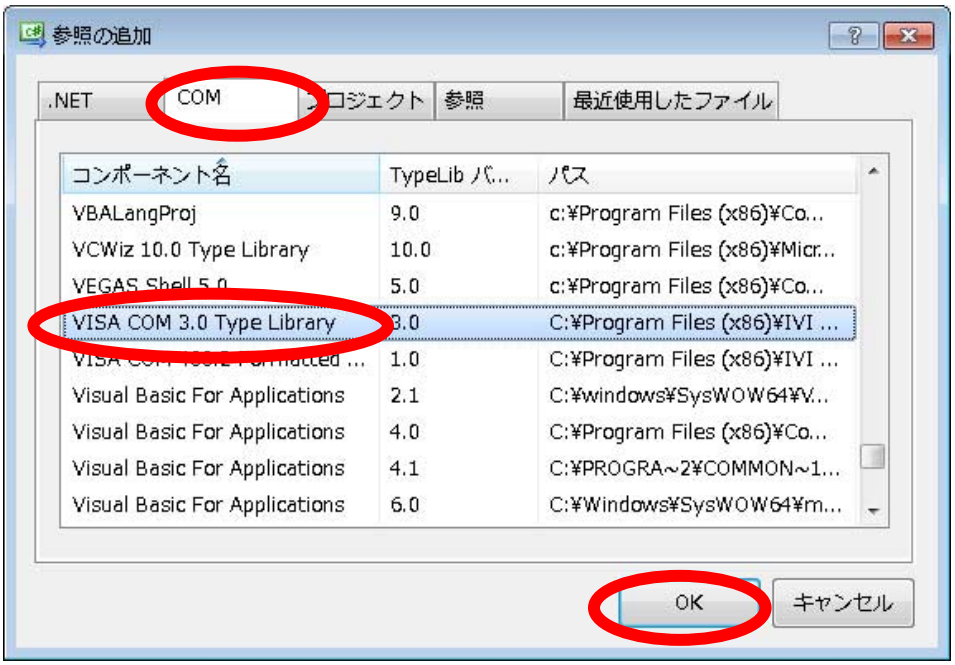


図 11 VISA COM 3.0 Library の追加(Visual C#)

⑦ ソリューション エクスプローラの Form1.cs を右クリックし、コードの表示を選択します。(図 12 参照)

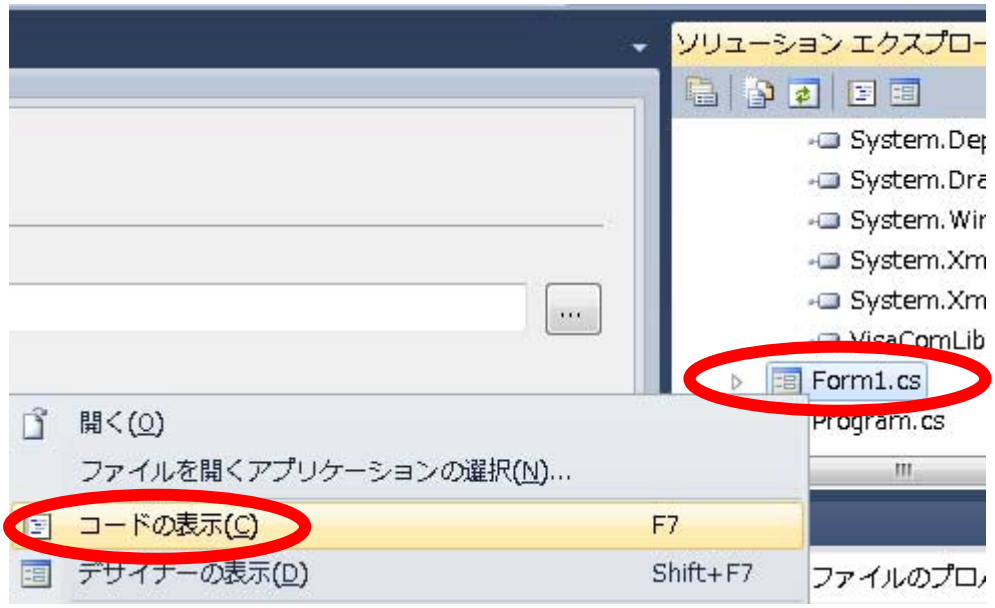


図 12 コードの表示

⑧ インポートする名前空間に「using Ivi.Visa.Interop;」を追加します。(図 13 参照)

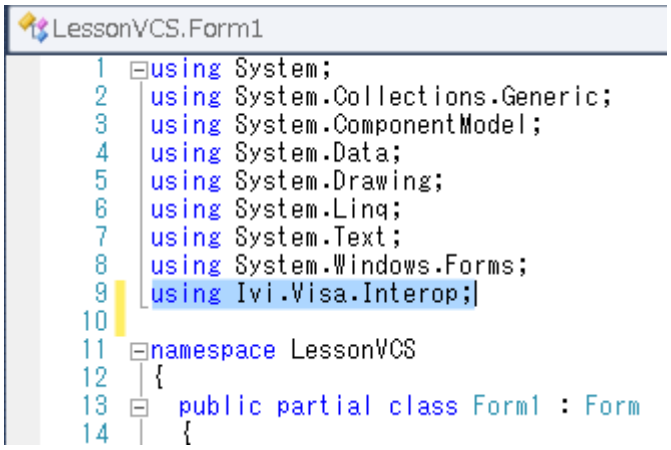


図 13 名前空間の追加(Visual C#)

ここまでで、プログラム上での下準備は、完了です。では、実際に、測定器に「*IDN?」を送り、その返り値を取得するコードを書いていきます。

３．３． 測定器との通信

① デザインウィンドウで、[表示] > [ツールボックス]で、ツールボックスを表示させ、ボタンコントロールを配置します。(図 14 参照)

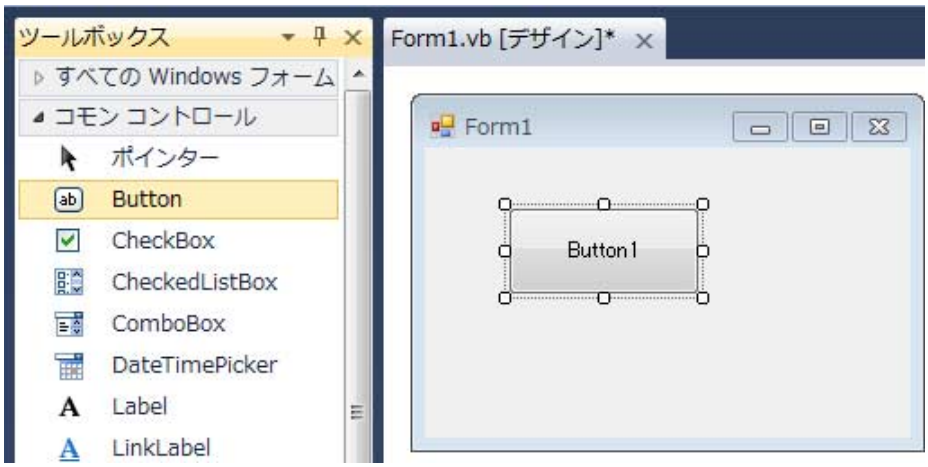


図 14 デザインウィンドウ(Visual Basic)

② 配置したボタンコントロールをダブルクリックし、コード画面を表示にします。

③ 以下のようにコードを追加します。

[VISA アドレス]の部分には、Agilent Connection Expert で確認した VISA アドレスを、以下のように入力してください。

例：rm.Open("USB0::0x0957::0x17A2::MY50500003::0::INSTR")

(ア) Visual Basic の場合

```
1  Public Class Form1
2
3      Private Sub Button1_Click(sender As System.Object, e As System.EventArgs) Handles Button1.Click
4          Dim rm As New ResourceManager()
5          Dim inst As New FormattedIO488()
6
7          ' 通信の開始
8          '[VISA アドレス]に接続した測定器のVISA アドレスを代入する
9          inst.IO = rm.Open("[VISA アドレス]")
10
11         ' 測定器のID確認
12         inst.WriteString("*IDN?")      ' IDクエリ
13         Dim str As String = inst.ReadString()  ' 応答の文字列取得
14
15         ' ダイアログボックスの表示
16         MessageBox.Show(Str)
17
18         ' 通信の終了
19         inst.IO.Close()
20
21     End Sub
22 End Class
23
```

ここを入力

図 15 コードの追加(Visual Basic)

----入力するコード----

```
Dim rm As New ResourceManager()
Dim inst As New FormattedIO488()
' 通信の開始
'[VISA アドレス]に接続した測定器の VISA アドレスを代入する
inst.IO = rm.Open("[VISA アドレス]")
```

' 測定器の ID 確認

inst.WriteString("*IDN?")　　' ID クエリ

Dim str As String = inst.ReadString()　' 応答の文字列取得

' ダイアログボックスの表示

MessageBox.Show(Str)

' 通信の終了

inst.IO.Close()

(イ) Visual C#の場合

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using Ivi.Visa.Interop;

namespace LessonVCS
{
  public partial class Form1 : Form
  {
    public Form1()
    {
      InitializeComponent();
    }

    private void button1_Click(object sender, EventArgs e)
    {
      ResourceManager rm = new ResourceManager();
      FormattedIO488 inst = new FormattedIO488();

      // 通信の開始
      inst.IO = rm.Open("[VISA アドレス]") as IMessage;

      // 測定器のID確認
      inst.WriteString("*IDN?");      // IDクエリ
      String str = inst.ReadString(); // 応答の文字列取得

      // ダイアログボックスの表示
      MessageBox.Show(str);

      // 通信の終了
      inst.IO.Close();
    }
  }
```

ここを入力

図 16 コードの追加(Visual C#)

----入力するコード-----

ResourceManager rm = new ResourceManager();

FormattedIO488 inst = new FormattedIO488();

```
// 通信の開始
//[VISA アドレス]に接続した測定器の VISA アドレスを代入する
inst.IO = rm.Open("[VISA アドレス]") as IMessage;
// 測定器の ID 確認
inst.WriteString("*IDN?");                    // ID クエリ
String str = inst.ReadString();  // 応答の文字列取得
// ダイアログボックスの表示
MessageBox.Show(str);
// 通信の終了
inst.IO.Close();
```

④ [F5]キーを押すかメニュから[デバッグ]>[デバッグ開始]を実行し、作成したウィンドウのボタンを押下します。(図 17 参照)

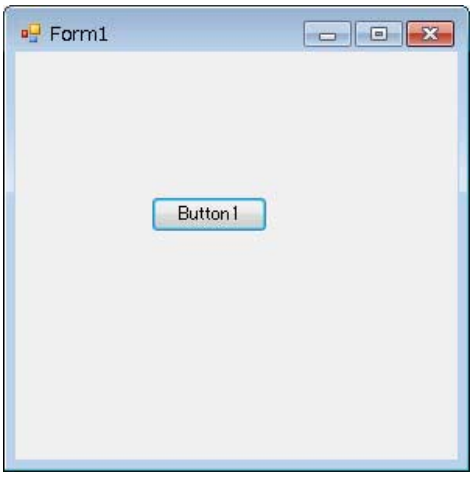


図 17 プログラムの実行

⑤ 図 18 のダイアログが表示され、測定器 ID を取得できます。

※ Agilent Technologies 社 MSO7000 シリーズオシロスコープの例

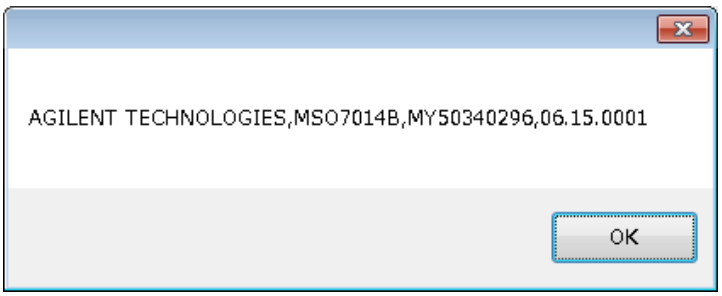


図 18 ID の取得

以上で、測定器から、測定器の ID を取得する事ができました。このように、測定器に対して、コマンドを送り、設定を行い、その返り値を取得するといった形で、制御プログラムは構成されます。

4. 制御プログラムの基礎知識

４．１． コマンド

測定器に対して、「*IDN?」という測定器の ID(モデル名)を問う文字列（コマンド）を送り、その応答を得る事ができました。「*IDN?」のように、測定器に、一定の動作をさせる、この文字列をコマンドと呼びます。一定の動作をさせるコマンドを複数組わせて送る事で、測定器を特定の状態にする、測定器に測定をさせる、測定値を取得するなどのリモート制御が実現できます。

また、特に、GPIB の仕様である IEEE488.2 にて定められているコマンドに、共通コマンドと、SCPI（スキッピ）コマンドがあります。

① 共通コマンド

共通コマンドとは、どの測定器を扱う上でも、同じ文字列を持つコマンドで、先の「*IDN?」や、他にも、測定器をリセット状態にする「*RST」などがあります。これらは、弊社の測定器でも、他メーカーの測定器でも、IEEE488.2 準拠の測定器であれば、お使いになることができます。

② SCPI（スキッピ）コマンド

SCPI コマンドは、測定に関するコマンドを規定したものです。コマンドは、階層構造（ツリー構造）となっており、上位（ルート）キーワードと下位のキーワードから構成され、コマンドの予測可能性を高めています。例えば、周波数を測定するコマンドは、「:MEASure:FREQuency?」、振幅を測定するコマンドは、「:MEASure:VPP?」となります。測定を司るコマンドは、どれもルートは同じ:MEASure となり、「:」（セミコロン）を付けて、下位キーワードとして、周波数測定、振幅測定を表します。これにより、“測定”コマンドを探す場合には、ルートが、:MEASure から始まるものを探せば、良いことになります。

現在、弊社の測定器のほとんどは、SCPI コマンド体系を採用しており、他社でも SCPI コマンド体系をサポートする測定器が増えています。

測定器のコマンドは、製品に付属されている取扱い説明書に記載されていることが一般的です。弊社では、ユーザーズガイド(User’s Guide)や、プログラマーズガイド(Programmer’s Guide)に、コマンドを記載しています。また、弊社では、これらのガイドは、Web サイトにて公開させて頂いております。

４．２． ライブラリ

４．２．１． ライブラリとは

プログラムから測定器と通信を行う際には、「ライブラリの参照」で追加し、必要に応じて、関数を呼び出して、通信を行う必要があります。

例えば、先ほど使った WriteString は、これらの物理インターフェースを介して、測定器に ASCII ベースのメッセージを送る役目を持っています。また、ReadString は、逆に、測定器からの ASCII ベースのメッセージを受信する役目を持っています。文字列以外に数値を受信する場合には ReadNumber、IEEE488.2 で規定されているバイナリのブロックデータを受信する場合には ReadIEEEBlock を使う等、その役割に応じた関数が用意されており、それらを必要に応じて使い分け、通信する必要があります。

これらの関数は、ライブラリを参照する事で、ご利用いただく事ができます。ライブラリは、測定器メーカーによって、個々用意されておりますが、弊社では、主に、SICL、VISA、VISA-COM というライブラリを提供しています。これらは、いずれも、IO Libraries Suite をインストールしていただくことでお使いいただく事が出来ます。

なお、ライブラリの詳しい取扱いに関しては、IO Libraries Suite をインストールして頂いた後に
[すべてのプログラム] > [Agilent IO Libraries Suite] >[Documentation]
にある Help をご参照ください。

４．２．２． ライブラリの利点

今回、使っていただいているものは、VISA-COM というライブラリです。VISA とは、Virtual Instrument Software Architecture の略で、IVI Foundation で制定された測定器制御のためのライブラリです。VISA は、測定器業界での共通規格であり、Agilent 以外の測定器メーカーからも提供されています。Agilent の VISA を呼び出すプログラムは、簡単な変更、もしくは、変更なしで、他メーカーの VISA を呼び出す事が可能です。例えば、Agilent の GPIB ボードを使って、作成されたプログラムであっても、VISA をサポートしている他メーカーの GPIB ボードをお使いいただく事が可能です。

また、VISA は、測定器で用いられる物理インターフェース（GPIB、RS-232、USB、LAN など）関係なく、同じプログラミング手法でアクセスを可能としています。例えば、測定器が、GPIB、USB、LAN の 3 つのインターフェースをサポートしている場合、GPIB ボードをベースに作成された制御プログラムであっても、アドレスの変更のみで、USB や LAN 経由での制御プログラムとしてお使いになる事ができます。

[まとめ]

- VISA、VISA-COM は、測定器業界での共通規格である。
- 他メーカーの VISA、VISA-COM を使う場合も、簡単な変更もしくは変更なしで使う事ができる。
- 物理インターフェースが変わっても、同じプログラムで制御することができる。

5. サンプルプログラム

ここからは、Agilent 製オシロスコープ（InfiniiVision シリーズ）のサンプルプログラムに関して、ご説明していきます。InfiniiVision シリーズ オシロスコープのサンプルプログラムは、下記よりダウンロードが可能となっております。

[ダウンロードサイト]

http://www.agilent.co.jp/find/InfiniiVision-sample

※ InfiniiVision シリーズ

DSOX/MSOX2000,3000、DSO5000、DSO/MSO6000、DSO/MSO7000 の各シリーズ

５．１． サンプルプログラムの説明

このサンプルプログラムは、コードだけではなく、既にデザインウィンドウに、測定を実施するためのボタンや、設定のパラメーター選択のためのドロップダウンメニューが配置されており、そのままでも、お使いになることができます。

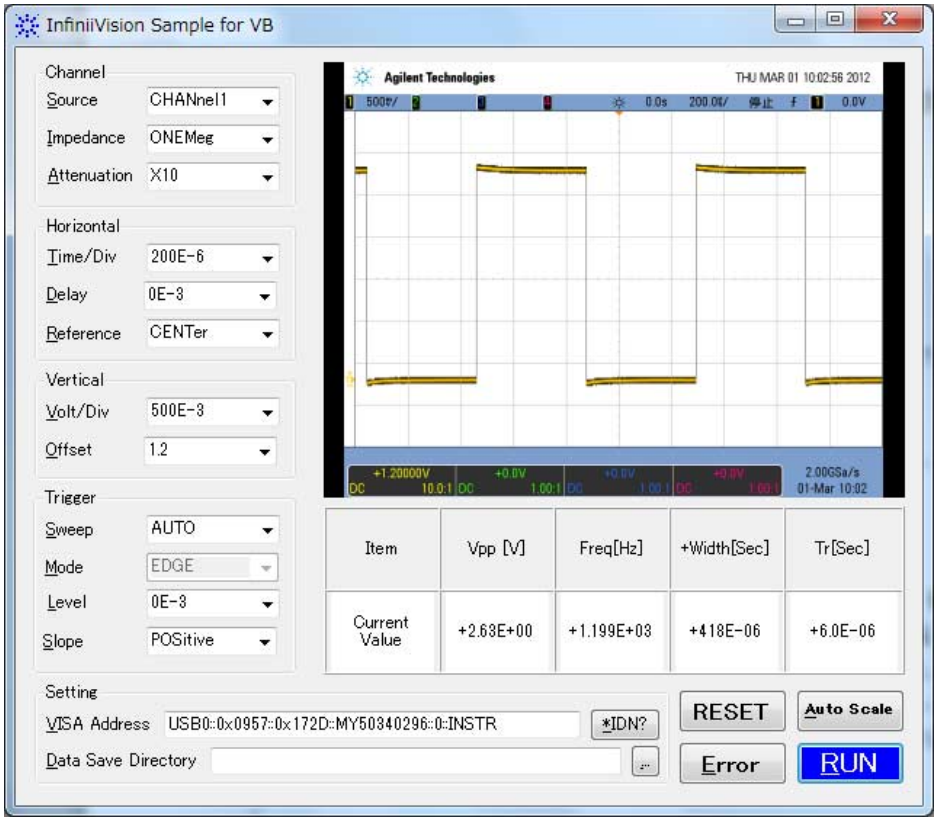


図 19 サンプルプログラム

このサンプルプログラムでは、オシロスコープの各パラメーター（時間レンジ、振幅レンジ、チャンネル、トリガ）を設定し、波形を取得できます。波形取得後は、振幅、周波数などの項目を測定、

オシロ画面をキャプチャし、ソフト上に表示させます。また、オシロ画面、取得した波形データは、指定したディレクトリに自動的に保存されます。リセット、エラー取得、Auto Scale を個別に実施させることができます。

５．２． サンプルプログラムの操作方法

解凍したフォルダより、以下を実施します。

(ア) Visual Basic の場合

¥InfiniiVisionSampleVB¥bin¥Release¥ InfiniiVisionSampleVB.exe

(イ) Visual C#の場合

¥InfiniiVisionSampleVCS¥bin¥Release¥InfiniiVisionSampleVCS.exe

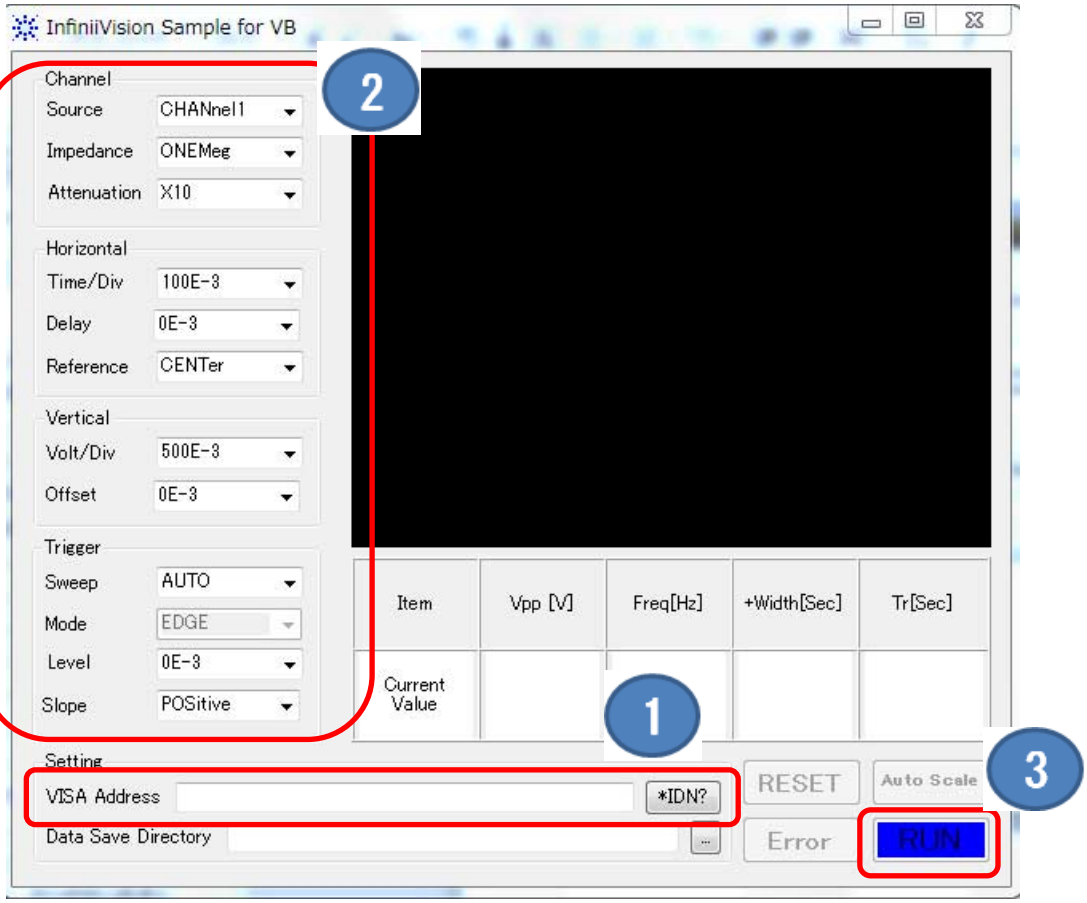


図 20 サンプルプログラムの操作方法

① VISA Address 欄に、制御するオシロスコープの VISA アドレスを入力し、*IDN?ボタンを押します。
オシロスコープの VISA アドレスは、前述の Agilent Connection Expert にて確認できます。
例： USB0::0x0957::0x17A2::MY50500003::0::INSTR
*IDN?ボタンを押すと、測定器のモデル名が表示されます。表示されない場合には、VISA アド

レスを再確認してください。また、Agilent Connection Expert にて、測定器が接続されているか確認してください。

② 入力している波形に合わせて、チャンネル、水平軸、垂直軸、トリガの設定を行います。

③ RUN ボタンを押すと、測定が開始されます。測定後、プログラム画面に、オシロスコープの画面画像が表示され、測定値が表示されます。画面画像、波形データは、Data Save Directory 欄で指定したディレクトリに保存されます。

なお、入力されている波形に対して、適切な設定がなされていない場合には、オシロスコープに波形が表示されないことや、トリガがかからずに、プログラムが停止することがあります。適切な設定に変更し、実施してください。

５．３． サンプルプログラムのコードの確認、カスタマイズ

サンプルプログラムのコードを確認、カスタマイズする場合には、プログラミングソフトを起動し、[ファイル] > [プロジェクトを開く] から、以下のプロジェクトファイルを開いてください。

(ア) Visual Basic の場合

¥InfiniiVisionSampleVB

(イ) Visual C#の場合

¥InfiniiVisionSampleVCS

５．４． サンプルプログラムのコマンド

このサンプルプログラムには、表１のコマンドが使われています。末尾に「?」がついているコマンドは、クエリ・コマンドと呼び、返り値を要求するコマンドとなります。クエリ・コマンドを送った後は、返り値の読み込みが必要となります。なお、コマンドの詳細に関しましては、各オシロスコープの Programmer's Guide をご参照ください。

※英語版ガイドのみのご提供です。日本語版はありません。

[InfiniiVision Oscilloscopes Programmer's Guide]

DSOX/MSOX2000 Series Oscilloscopes Programmer's Guide

http://www.home.agilent.com/upload/cmc_upload/All/2000_series_prog_guide.pdf

DSOX/MSOX3000 Series Oscilloscopes Programmer's Guide

http://www.home.agilent.com/upload/cmc_upload/All/3000_series_prog_guide.pdf

DSO5000 Series Oscilloscopes Programmer's Guide

http://www.home.agilent.com/upload/cmc_upload/All/5000_series_prog_guide.pdf

DSO/MSO7000 Series Oscilloscopes Programmer's Guide

http://www.home.agilent.com/upload/cmc_upload/All/7000B_series_prog_guide.pdf

コマンド	解説
*IDN?	測定器の ID 取得
*RST	リセット状態にする
:AUToscale	オートスケール実施
:SYSTem:ERRor?	エラー・キュー内容取得
:CHANnel<n>:SCALe	縦軸のレンジ設定（1 マスあたりの電圧値）
:CHANnel<n>:OFFSet	縦軸のオフセット設定
:CHANnel<n>:DISPlay	チャンネルの表示オン/オフ
:CHANnel<n>:IMPedance	チャンネルの入力抵抗選択
:CHANnel<n>:PROBe	チャンネルのプローブ減衰比設定
:TIMebase:SCALe	横軸のレンジ設定（1 マスあたりの時間）
:TIMebase:POSition	横軸のポジション設定（Delay つまみ）
:TIMebase:REFerence	トリガポジション
:TRIGger:SWEep	自動トリガ、ノーマルトリガの切り替え
:TRIGger:MODE	トリガモードの選択（エッジ等）
:TRIGger:SOURce	トリガソースの選択
:TRIGger:EDGE:LEVel	トリガエッジのレベル
:TRIGger:SLOPe	トリガスロープ選択（立ち上がり or 立下り）
:STOP	収集停止
:SINGle	シングルトリガ
:TER?	トリガが掛かったかどうか調べる
:MEASure:VPP?	Vpp 値の取得
:MEASure:FREQuency?	周波数値の取得
:MEASure:PWIDth?	正のパルス値の取得
:MEASure:RISetime?	立ち上がり時間値の取得
:WAVeform:SOURce	転送波形の指定
:WAVeform:FORMat	波形転送時のフォーマット選択
:WAVeform:UNSigned	波形転送時の整数型選択
:WAVeform:BYTeorder	波形転送時のバイト・オーダー選択
:WAVeform:XINCrement?	サンプルポイントの時間間隔の取得
:WAVeform:XORigin?	横軸の基準位置の取得
:WAVeform:YINCrement?	サンプルポイントの電圧間隔の取得
:WAVeform:YORigin?	縦軸の基準位置取得
:WAVeform:DATA?	波形データの取得
:DISPlay:DATA? BMP	画面の画像データの取得

表 1. SCPI コマンド

6. オシロスコープ制御のポイント

このサンプルプログラムには、オシロスコープを制御する際に、良く利用されるポイントが、盛り込まれています。ここでは、そのポイントに関して解説します。

６．１． 制御プログラムの流れ

基本的に、測定器を制御する際のプログラムは、大きく、初期化、設定、測定（出力）に分かれます。これらを意識して、プログラムを作成することが大切です。また、設定では、同じルートに属するコマンドは 1 か所に集める事が望ましいです。これにより、プログラムがわかりやすくなるだけではなく、スループットの改善を図ることもできます。

よくある事として、同じ条件での測定を繰り返すのに、繰り返しの際に、初期化し、再度、同じ設定を実施、測定する事があります。これは、初期化や設定が無駄になります。同じ条件での繰り返し測定であれば、測定コマンドだけを送る事で、無駄な測定器の状態の変化を減らす事ができ、スループットを上げる事が出来ます。

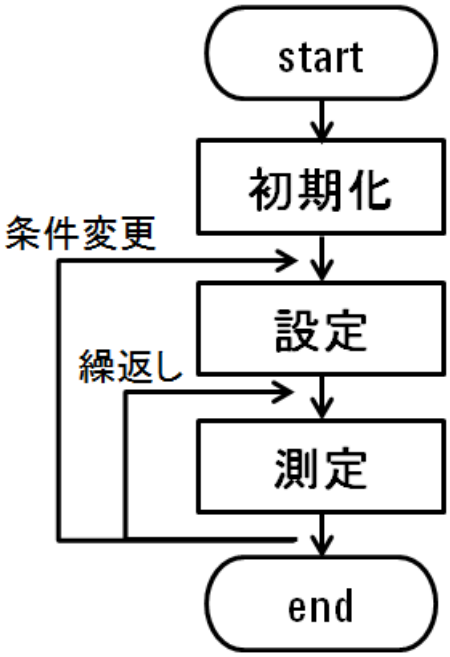


図 21 コマンドの流れ

設定後の Wait は十分ですか?

測定器によっては、コマンドの処理や動作が遅いものがあります。この場合、続けて、コマンドを送っても、設定が、きちんとされずに、次の動作に移ってしまう場合があります。Wait を入れるなど、余裕をもって、コマンドを送ることをお勧めします。また、「*OPC?」コマンド（Operation Complete）は、設定終了の確認に便利です。「*OPC?」 は、これまでの処理が完了した際、"1"を返すコマンド

です。一連の設定コマンドの最後に、「*OPC?」を配置し、"1"が返ったら、次の処理に移るようにします。処理や設定に時間が掛かる場合、"1"が返るまでに時間が掛かりますので、この場合は、タイムアウトを長めに設定します。また、測定器によっては、「*OPC?」は、処理が終了している場合には"1"を、していない場合には、"0"を返すものもあります。

６．２．トリガ待ち

今回は、ノーマルモードとなっているため、シングルトリガ「:SINGle」を送った後、トリガに設定した所定の信号が入力された時点で、トリガが掛かり、波形収集が行われる事になります。トリガが掛かり、画面に、波形が表示された後に、測定や波形データ転送のコマンドを送ります。この時、「:SINGle」の後に、そのまま測定コマンドを送ってしまうと、所定の信号が入らない場合、トリガをかける事ができず、測定できませんので、測定コマンドの部分で、タイムアウトが発生してしまいます。そのため、「:SINGle」を送った後に、所定の信号が入るまで、ウェイトを置くか、もしくは、オシロスコープに対し、トリガが掛かったかどうかを調べる必要があります。このサンプルでは、オシロスコープに対して、「:TER?」というトリガが掛かったかどうかを調べるコマンドを送っています。「:TER?」の返り値が 0 であれば、トリガが掛かっていませんので、再度、「:TER?」を送ります。返り値が１であれば、トリガは掛かっていますので、その次の測定に進むようにしています。

同じようなトリガ待ちを調べる方法として、Operation Status Condition Register の bit3（RUN）を読む方法があります。

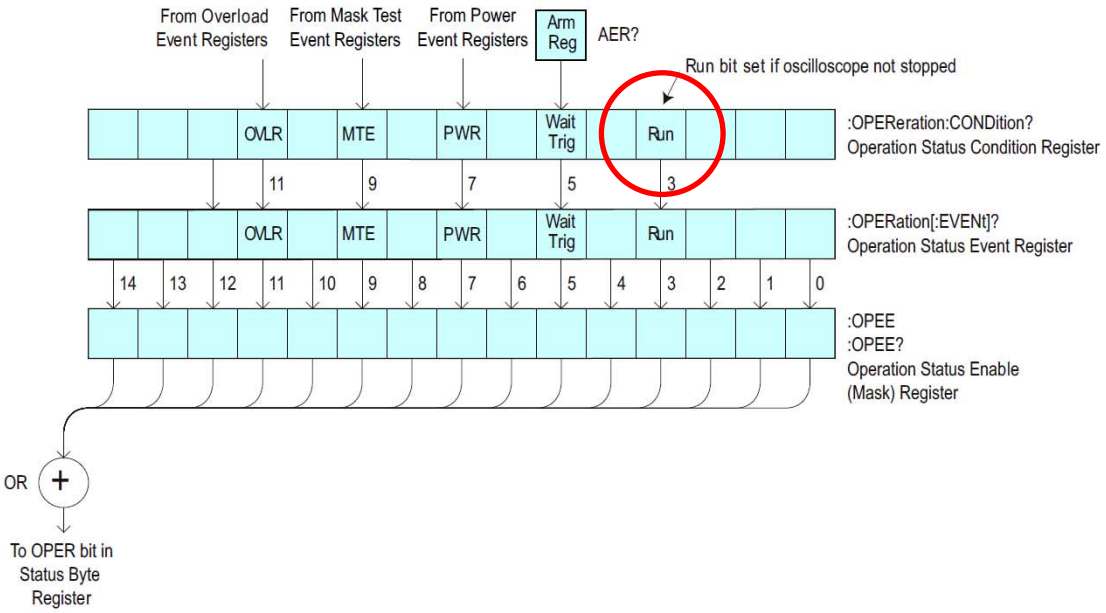


図 22 Operation Status Condition Register

レジスタは、測定器の状態を表す役割を持っており、レジスタの状態を調べる事によって、その測定器の状態を知る事ができます。弊社のオシロスコープには、機能別に、いくつかレジスタがありますが、その中で、Operation Status Condition Register というレジスタには、Run という項目があります。これは、オシロが Run 状態であれば、その Run のビットが立っており、Stop 状態であれば、その Run のビットは下がっている状態を示します。レジスタのビット状態を調べるコマンドとして、「:OPERegister:CONDition?」があります。このクエリ・コマンドの返り値は、2 進数で重み付けられた 10 進数となります。例えば、8 であれば、2^3 で、3bit に位置するビットが立っている事になります。また、例えば、12 であれば、2^2 + 2^3 で、2bit と 3bit に位置するビットが立っている事になります。

Run の項目は bit3 になります。そのため、「:OPERegister:CONDition?」の返り値に対して、論理演算を使って、

演算 （B の bit3 以外を Off にして A に代入する例）

```
A = B AND &H08     ' VB
A = B & 0x08;            // C#
```

Visual Basic での比較方法

```
if ((B AND &H8) <> 0)              ' flag on（Running。再度、レジスタを調べる）
if ((B AND &H8) == 0)              ' flag off（Stop。波形補足終了。次の動作へ）
```

Visual C#での比較方法

```
if( (B & 0x08) <> 0)                // flag on（Running。再度、レジスタを調べる）
if( (B & 0x08) == 0)                // flag off（Stop。波形補足終了。次の動作へ）
```

と処理することで、トリガ待ちの状態を調べる事ができます。
トリガ待ち状態を調べるには、「:TER?」、Operation Status Condition Register のどちらでも、好きな方法をお使い頂ければ結構です。

６．３． 波形のデータ転送

波形のデータ転送の際には、いくつか注意すべき点があります。ここでは、そのポイントに関して、説明をします。

６．３．１． ASCII 転送とバイナリ転送

「:WAVeform:DATA?」を送り、オシロスコープから波形データを転送する際、転送コーディングとして、ASCII、バイナリでのいずれかを選択する事ができます。ASCII を選択した場合、測定器は、ASCII コード（128 種類のローマ字、数字、記号、制御コード）を使って、データを転送します。バイナリを選択した場合、バイト（8bit）もしくはワード（16bit）にてデータを転送します。

ASCII でのデータ転送は、そのままの電圧値が ASCII としてデータ転送されるため、プログラムで、後で、電圧値に変換せずに済みますが、そのデータ転送量は膨大となり、大量のデータを受信するには向きません。また、バイナリ転送の場合には、電圧値がバイナリデータとして転送されるために、プログラムで受信後、電圧値に変換する必要がありますが、ASCII に比べて、転送量を抑える事ができ、転送時間を短くすることができます。 今回のサンプルプログラムでは、バイナリ転送を用いています。ASCII、バイナリの選択は、「:WAVeform:FORMat」 コマンドで実施できます。

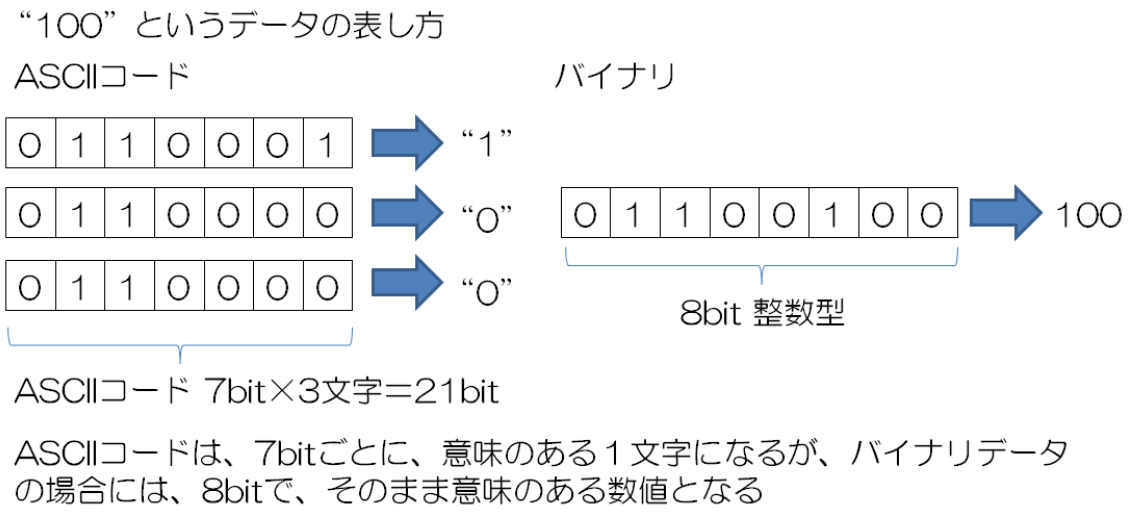


図 23 ASCII とバイナリのデータ量の比較

６．３．２． バイナリブロックデータ転送

今回のサンプルプログラムでは、バイナリ転送を用いて、波形の数値データの転送を行っています。なお、オシロスコープからの転送には、IEEE488.2 にて規定されているバイナリブロックデータ転送形式が用いられています。バイナリブロックデータ転送形式では、ヘッダ + データという形で転送されます。以下に例を示します。

バイナリブロックデータ転送（例）:

#800000200<DAB>...<DAB><DAB><DAB><LF>

#:データブロックの開始コード
8:後続する桁数。00000200 の桁数を示す。
00000200:後続するバイナリデータの数を示す。この例では 200 バイトが続くという意味。
DAB:データ（バイナリ）
<LF>:バイナリデータの最後を示す終端コード
→ #800000200 までがヘッダ部分となり、DAB にて、実際のデータ転送がなされます

VISA-COM をライブラリとしてお使い頂いている場合には、ReadIEEEBlock 関数を使う事で、ヘッダ部分を処理した上で、データのみが読み取られます。そのため、VISA-COM をお使い頂いている場合には、ReadIEEEBlock 関数をお使い下さい。他のライブラリを使っている場合には、まず、IEEE488.2 のブロックデータ転送に対応している関数があるかを調べ、無い場合には、ヘッダ部分のみを除去し、バイナリデータ部分だけを読み取るように工夫する必要があります。

６．３．３． バイト、ワード

今回、サンプルプログラムでは、ワードにて、データ転送をしています。ワードとは、一つの電圧値を 2 バイト分（＝16bit）のデータで表すことです。1 バイト分（=8bit）のデータで表す事は、バイトと呼びます。通常、オシロスコープは、8bit 分解能であるために、ノーマルモードで値の取得を実施している場合には、バイトで、8bit の分解能を表現でき、ワードの必要はありません。このような場合には、バイトを転送として選択する方が、転送容量を少なくできる分、転送時間を早くする事ができます。ただし、オシロスコープで、アベレージモード、高分解能モードを使用している場合には、分解能が 8bit より増えますので、バイトでは分解能の表現ができず、必ず、ワードで転送する必要があります。ワード、バイトの切り替えは、「:WAVeform:FORMat」 コマンドで実施できます。

６．３．４． 符号付数値表現

InfiniiVision シリーズでは、データ転送の際、データの数値表現方法として、符号付もしくは符号なしを選択することができます。デフォルトでは、符号なし（Unsigned Integer）となっており、「:WAVeform:UNSigned」コマンドで、符号付（Signed Integer）にも変更する事ができます。今回のプログラムでは、符号付として、取得しています。

６．３．５． データ転送時のバイト順

バイナリデータ転送の際、ワード（2 バイト）を選択する場合は、上位バイトと、下位バイトのどちらを先に転送するのか決めておく必要があります。バイト転送時に、下位バイト（Least Significant Byte）を先に転送する事を LSB ファースト/ビックインディアン、上位バイト（Most Significant Byte）から先に転送する事を MSB ファースト/リトルインディアンと呼びます。

受信側の PC にて、メモリにそのまま展開した際、送信側（測定器）が転送するバイト順序と同じであれば、正しい数値として扱えますが、同じではない場合、正しい数値として扱うには、上位バイトと下位バイトを入れ替える必要があります。

通常、インテル系のアーキテクチャーでは、LSB ファーストでメモリへの展開を実施します。また、InfiniiVision シリーズも、デフォルトは、LSB ファーストにて転送が実施されますので、順序を変更する必要はありません。なお、InfiniiVision シリーズのバイト順序を変更する場合には、「:WAVeform:BYTeorder」 コマンドを用います。

６．３．６． データ読み取り時のタイムアウト設定（よくあるエラー）

データを読み取る際、転送データが多いと、転送、読み取りに時間が掛かるために、読み取り部分で、タイムアウトが発生する場合があります。タイムアウトとは、動作に期待している時間より長くかかる場合に、途中で打ち切って止める行為です。読み取り部分でタイムアウトエラーが発生する場合、タイムアウトの値を大きく設定することをお勧めします。今回のサンプルプログラムでは、データの読み取り前に、10 秒にタイムアウトの設定を変更しています。

```
inst.IO.Timeout = 10000
```

特に、波形の数値データが大きくなる場合や、画面の画像データを転送する場合には、タイムアウト設定に注意してください。

６．４． 取得したバイナリデータの電圧値への変換

バイナリデータとして取得した波形データは、そのまま状態では、オシロスコープの縦軸の分解能を示しています。電圧値ではありません。そのため、電圧値として読むには、プリアンブルと呼ばれる値を取出し、その上で変換する必要があります。プリアンブルデータは、以下のコマンドで取得する事ができます。

Y 軸のインクリメントデータ

「:WAVeform:YINCrement?」

Y 軸の原点データ

「:WAVeform:YORigin?」

上記のデータを使って、バイナリデータから電圧値へは、以下のように変換します。

符号付で取得した場合

電圧値= [インクリメント] × [符号付データ] + [原点]

符号なしで取得した場合

電圧値= [インクリメント] × [符号なしデータ － Yref] + [原点]

※ Yref: 固定値。バイトでの取得時は 128、ワードでの取得時は 32768

今回のサンプルプログラムでは、符号付き数値表現（Signed Integer）にて取得しているため、電圧値= [インクリメント] × [符号付データ] + [原点]を用いて、変換しています。

６．５． デバッグ

プログラムを作成しても、望んだように動かない場合、デバッグ作業をする必要があります。しかし、ただ、闇雲に、デバッグ作業を実施すると、時間が掛かる場合があります。以下の視点でデバッグしていただくと、作業時間を短くできると思います。

1) 問題の部分だけを抜き出す、プログラムを簡素化する

大きいプログラムになればなるほど、プログラマーが意図しない動きが混入する場合があります。そのため、まずは、問題の部分だけを抜き出したプログラムにすることが肝要です。また、複数の測定器を同時に動かす制御プログラムの場合には、問題の測定器のみにすることも有効です。ループや条件分岐が入り組んだプログラムの場合には、ループや条件分岐を止めてみると良いでしょう。問題部分の絞り込みを実施することで、デバッグ作業を楽にすることができます。

2) ステップ実行で 1 行 1 行実施してみる

問題部分を絞り込んだ後は、プログラムを、ステップ実行で動作させ、明確に問題箇所を特定します。1 行毎に、ステップ実行をさせながら、測定器の動作と、プログラムへの返り値を確認していきます。よくある事として、

（良くある例 1）コマンドを送ったのに、設定がされない!?

→ コマンドの記述ミス

→ Undefined Header , Syntax Error＝コマンドの記載ミス

→ コマンドの記述ミスは、:SYST:ERR? で確認

→ 他に必要なコマンドが送られていない

→ コマンドの送る順番

→ 設定するのに、決められた順番で、コマンドを送る必要がある

→ 他のコマンドで設定が変更されてしまっている

→ タイミングミス、測定器の動作が追い付いていない

→ wait や *OPC?を入れ、タイミングを調整する

（良くある例 2）タイムアウトエラーが発生してしまう!?

→ 測定器との接続がされていない

→ Connection Expert で測定器との通信ができるか再確認

→ 動作/処理に時間が掛かるのにタイムアウトが短い

（例：長い波形データの取り込みをしているのに、タイムアウト時間の設定値が短い）

がありますが、ステップ実行により、明らかにしていくことができます。

3) コマンドを送るタイミングを疑う

ステップ実行で、問題がない場合には、コマンドを送るタイミングを疑います。例えば、測定器の設定直後に、測定を開始する場合には、測定器によっては、ある程度時間が経たないと、安定しない場合があります。このような場合は、ステップ実行では、逆に、上手く Wait が入ってしまうために、問題が発生しなくなることがあります。設定コマンドを送った後や、トリガが掛かった後など、タイミングが疑われる部分に、十分な Wait を入れ、動作を確認します。

4) 測定器のバグと思われる場合には、ファームウェアを更新する

測定器の動作が、プログラマーズガイドの説明通りではない場合、測定器のバグであることがあります。バグは、ファームウェアの更新で、修正されている場合があります。アジレントでは、最新のファームウェアを Web サイトで公開しており、すぐに、ユーザーにてファームウェアの更新ができるようになっています。更新後も、現象が再現する場合には、サポート窓口までお問い合わせください。その際、問題個所を特定したプログラム、コードと、使っている測定器、ファームウェアバージョンをご連絡ください。

5) IO モニタの活用

アジレントでは、ツールとして、IO モニタをご用意させて頂いています。IO モニタとは、測定器に出力したコマンド、測定器から読み込んだデータ、IO のタイミングなどを簡単に確認するためのツールです。これにより、予期せぬ返り値が返ってきている事や、実際に予想していたようなコマンドの送り方、タイミングになっていないことを確認することができます。

① Agilent Connection Expert を起動します。スタートボタンから [スタートボタン]>[Agilent IO Libraries Suite]>[Agilent Connection Expert]を実行します。(エラー! 参照元が見つかりません。参照)

② 起動後、メニュより[Tools]>[IO Monitor]を選択します。()

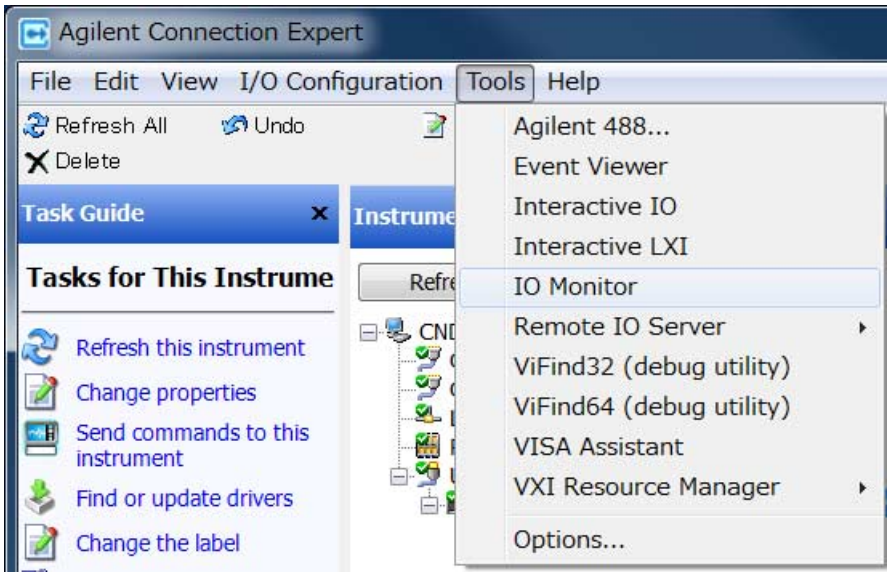


図 24 IO Monitor の起動

③ IO Monitor のメニュより [Tools]>[Options]を選択し、IO Monitor Options を表示させます。

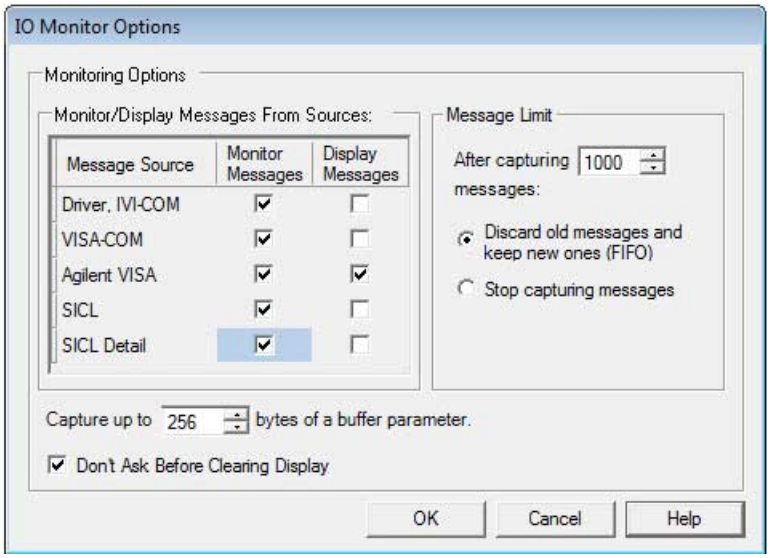


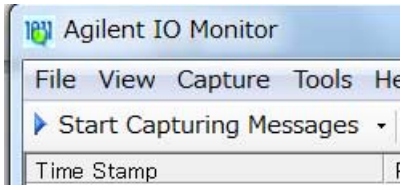
図 25 IO Monitor Option の画面

Monitoring Option では、モニタするレイヤ（Monitor Message）、画面に表示するレイヤ（Display Message）を選択します。ライブラリは、上位、下位のレイヤに分かれており、上位レイヤ（VISA-COM）を指定すると、最も、アプリケーションに近いレイヤをモニタすることを意味し、下位レイヤ（SICL Detail）を指定すると、最も、ハードウェアに近いレイヤをモニタする事を意味します。

Message Limit では、モニタするメッセージ数を指定します。100 から 10000 メッセージまでを指定できます。

Capture up to...bytes of a Buffer parameter では、記録するバイト数を指定します。32 から 65536 バイトまでを指定できます。

④ 実際に記録を開始するには、Start Capturing Messages ボタンを押します。



⑤ 下記は、USB にて接続したオシロスコープと PC とのやり取りを IO Monitor で観測した例です。

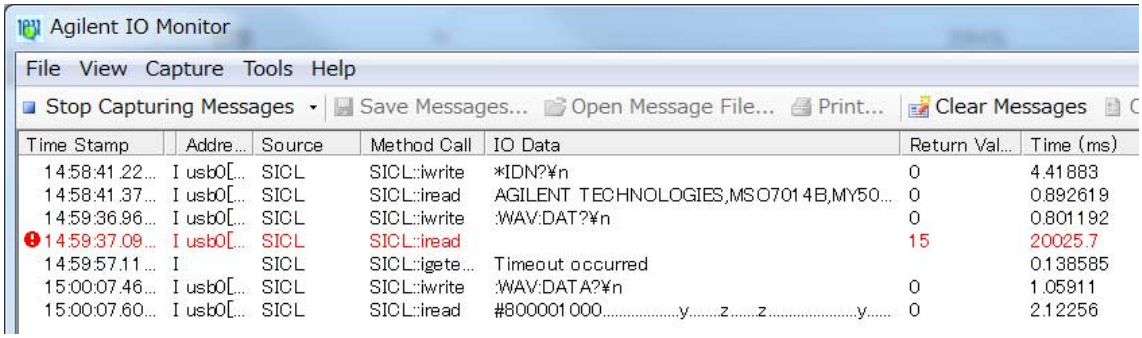


図 26 IO Monitor Option の観測例

1，2 行目：*IDN?¥n （¥n は、コマンドのデリミタ） が送られ、その返り値として、測定値の名前が返ってきている事が分ります。

3-5 行目： :WAV:DAT?¥n という間違ったコマンド（最後の A が足りない）が送られ、その返り値を求めていますが、測定器が返り値を出さなかったために、タイムアウトになっています。そのため、4 行目は、赤い文字になり、5 行目で、タイムアウトが発生している事を表示しています。

6,7 行目: :WAV:DATA?¥n という正しいクエリコマンド（波形データを求めるコマンド）が送られ、その返り値として、#800001000 から始まるバイナリデータ（バイナリブロックデータ転送）が返ってきている事が分ります。

IO Monitor は、強力なデバッグツールの一つです。デバッグ原因の特定などに活用する事ができますので、デバッグの際には、是非、ご利用下さい。

7. 測定器プログラムのサポート

このガイドやサンプルプログラムに関しては、下記まで、お問い合わせください。

アジレント・テクノロジー株式会社
電子計測本部・計測お客様窓口
フリーダイアル： 0120-421-345
Email ： contact_japan@agilent.com

計測お客様窓口では、このガイドに対する質問、サンプルプログラムに関してのインストール方法、実行方法、プログラム内容等のお問い合わせを承っております。お問い合わせの際は、Windows のバージョン、言語のバージョン、ご利用の測定器モデル名、具体的なお問い合わせ内容をお知らせ下さい。その他、Agilent 社製測定器のコマンドや動作に関する説明、他社製測定器から弊社測定器へのポーティング、IO Libraries Suite による測定器の制御など、弊社製品に関してのご質問、ご相談も承っております。

新規プログラム作成のご依頼や、サンプルプログラムのカスタマイズのご依頼、お客様が作成・カスタマイズなされたプログラムのデバッグ作業は承っておりません。

Visual Studio 2010 Express Edition の使い方や文法など、プログラム開発環境やプログラム言語の一般的なご質問に関しましては、サポートしておりません。一般の書籍または web、Help 等の情報を、ご確認いただきますようお願いいたします。

【著作権および免責事項】
著作権はアジレント・テクノロジー株式会社が保有しています。このガイド、ソフトウェアを使用したことによって生じたすべての障害・損害・不具合等(含、ソフトウェアのバグ)に関しては、弊社および弊社の所属するいかなる団体・組織とも、一切の責任を負いません。各自の責任においてご使用ください。
【転載条件】
無断での転載・配布はご遠慮下さい。

自動計測関連情報

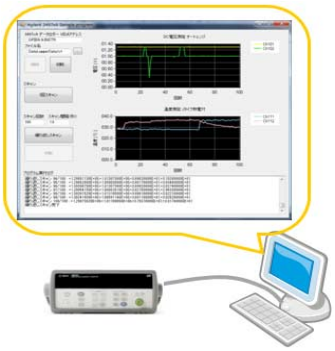
【ソフトウェア】

[IO Libraries Suite 最新版]
IO Libraries Suite は VISA 等を含むコネクティビティ・ソフトウェアです
http://www.agilent.com/find/iolib
2012 年 11 月現在の最新版は 16.3 です。バージョン 16.X は無償でご利用いただけます。

Agilent IO Libraries Suite 16.3/16.2 簡易取扱説明書]
http://cp.literature.agilent.com/litweb/pdf/5991-0331JAJP.pdf

[Agilent VEE Pro 最新版] (計測制御用のプログラム開発環境です)
http://www.agilent.com/find/vee
VEE Pro は有償です。上記より 30 日間の評価版をダウンロード可能です。
2012 年 11 月現在の最新版は 9.3 です。

【サンプル・プログラム (無償) 】



サンプル・プログラム、日本語資料のダウンロードが可能です

[Agilent PS-X30 G10101A E3600 DC 電源 サンプルプログラム]
http://www.agilent.co.jp/find/E3600_sample_program

[Agilent PS-X30 G10102A 34970A/34972A データロガー サンプルプログラム]
http://www.agilent.co.jp/find/datalog_sample_program

[InfiniiVision オシロスコープ用サンプル・プログラム ライブラリ]
http://www.agilent.co.jp/find/InfiniiVision-sample

【定期開催トレーニング】

【特長】

ポイント1 **受講者の高い満足度**

受講者の50%以上が上司/同僚からの推薦。
さらに、受講後アンケートで88%が「他の方に勧めたい」と回答

ポイント2 **少人数制。1人1台での測定器実機実習**

「実機演習有り」のコースでは、測定器を受講者1人1台使用できますので、自分のペースで実習できます。
「実機演習無し」のコースでも、少人数制ですので、わからない点は講師に質問しやすい環境

ポイント3 **お客様先でのトレーニング実施もお勧め**

定期開催以外にも、経験豊富な講師が、御社にてトレーニングを実施いたします。新人研修などにご利用ください

ポイント4 **受講後のアフターサポートも安心**

計測機器ベンダーサポートランキングで、7 年間連続の総合 No1 獲得。
専任エンジニアが常駐し、受講内容・測定方法のアドバイスを提供

[トレーニングコース一覧](日程も記載しています)
http://www.agilent.co.jp/FIND/training

[自動計測トレーニングコース詳細]
http://www.agilent.co.jp/find/programing105
[Excel で始める測定器制御]
[Visual Basic.net による測定器制御 - VISA COM ライブラリ編 -]
[Excel® VBA による測定器自動制御入門 オシロスコープ編]
[VEE 9.3 基礎]
[VEE 9.3 プログラミング]

アジレント・テクノロジー株式会社
本社〒192-8510 東京都八王子市高倉町 9-1

計測お客様窓口
受付時間 9:00-18:00(土・日・祭日を除く)
TEL ■■ 0120-421-345
(042-656-7832)
FAX■■ 0120-421-678
(042-656-7840)
Email contact_japan@agilent.com

電子計測ホームページ
www.agilent.co.jp

記載事項は変更になる場合があります。
ご発注の際にご確認ください。

©Agilent Technologies. Inc. 2013

Published in Japan, February 14 ,2013
5991-0244JAJP
0000-08A

Agilent Technologies